Measurement Algorithm Variation

Prior Solutions

Most prior solutions to this problem are either to provide the source code to a measurement algorithm or to break the algorithm down into small "chunks" to be recombined by the user in the context of the variation needed.

The disadvantages of the source code approach are the loss of proprietary knowledge from making the code public and the support problem caused by defects introduced from user modification of the code.

The disadvantage or the "chunking" approach is the documentation and training required for the user to take advantage of the measurement code.

Approaches used in other domains include inheritance, extensions, parameterization, templates, code generators, design patterns, and plug-ins. All of these except plug-ins require user access to and modification of the source code and therefore the proprietary algorithm. A Plug-in is one approach to solving the measurement problem, but it has never been applied in this context.

Problems Solved by This Invention

Agilent provides measurement knowledge for our customers in the form of reusable code modules. This measurement knowledge is valuable and it often embodies proprietary knowledge. Unfortunately not all application conditions can be anticipated when the code is written. Therefore, the customer must have the ability to affect some parts of the measurement algorithm. For instance, a new device may have an unexpectedly slow response time under some conditions. Delays may have to be inserted in the measurement algorithm at certain points to make the device work correctly. Another problem is that Device Under Test (DUT) state may have to be coordinated with the measurement algorithm state. Traditionally, we have had to provide the source code to the user to allow them to change or augment the algorithms.

This invention provides means for Agilent to open parts of the measurement algorithm to our users for optional modification without having to show source code or document all implementation details. This gives the users the ability to easily modify certain aspects of the behavior without a complex programming task. It also preserves the main functional behavior of the measurement algorithm. That is, it does not have to be re tested or characterized to the same level of detail that would be required if the user directly modified the core measurement algorithm.

Advantages over Prior Solutions

The source code remains proprietary to Agilent.

Users have the ability to modify selective behavior without invalidating the measurement algorithm.

Users can modify some behavior without having to understand the entire algorithm.

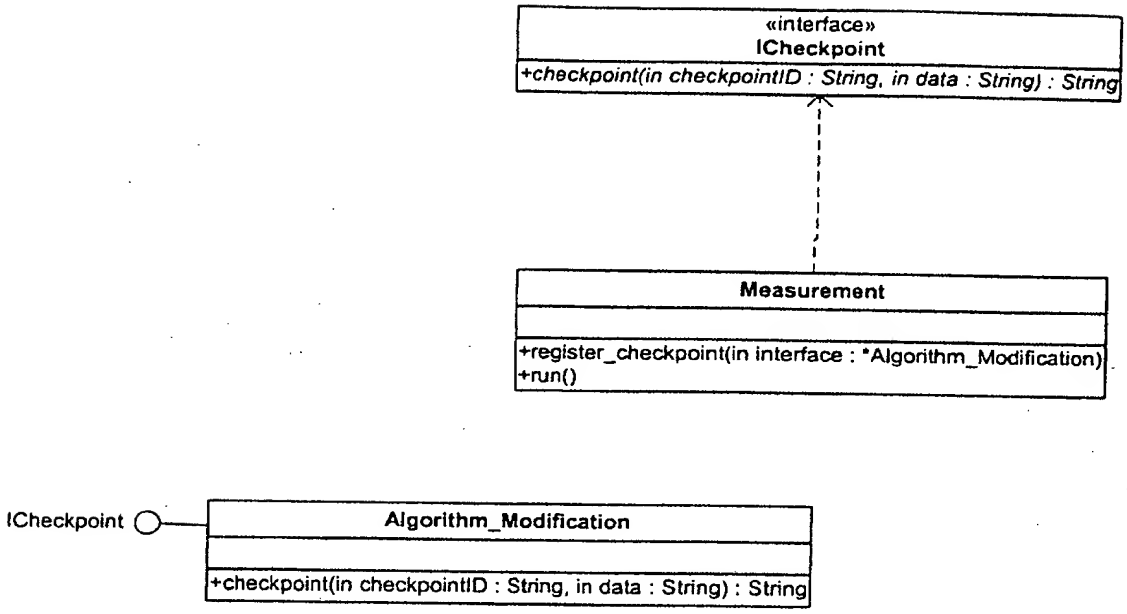
The user's customization ability is easier to document.

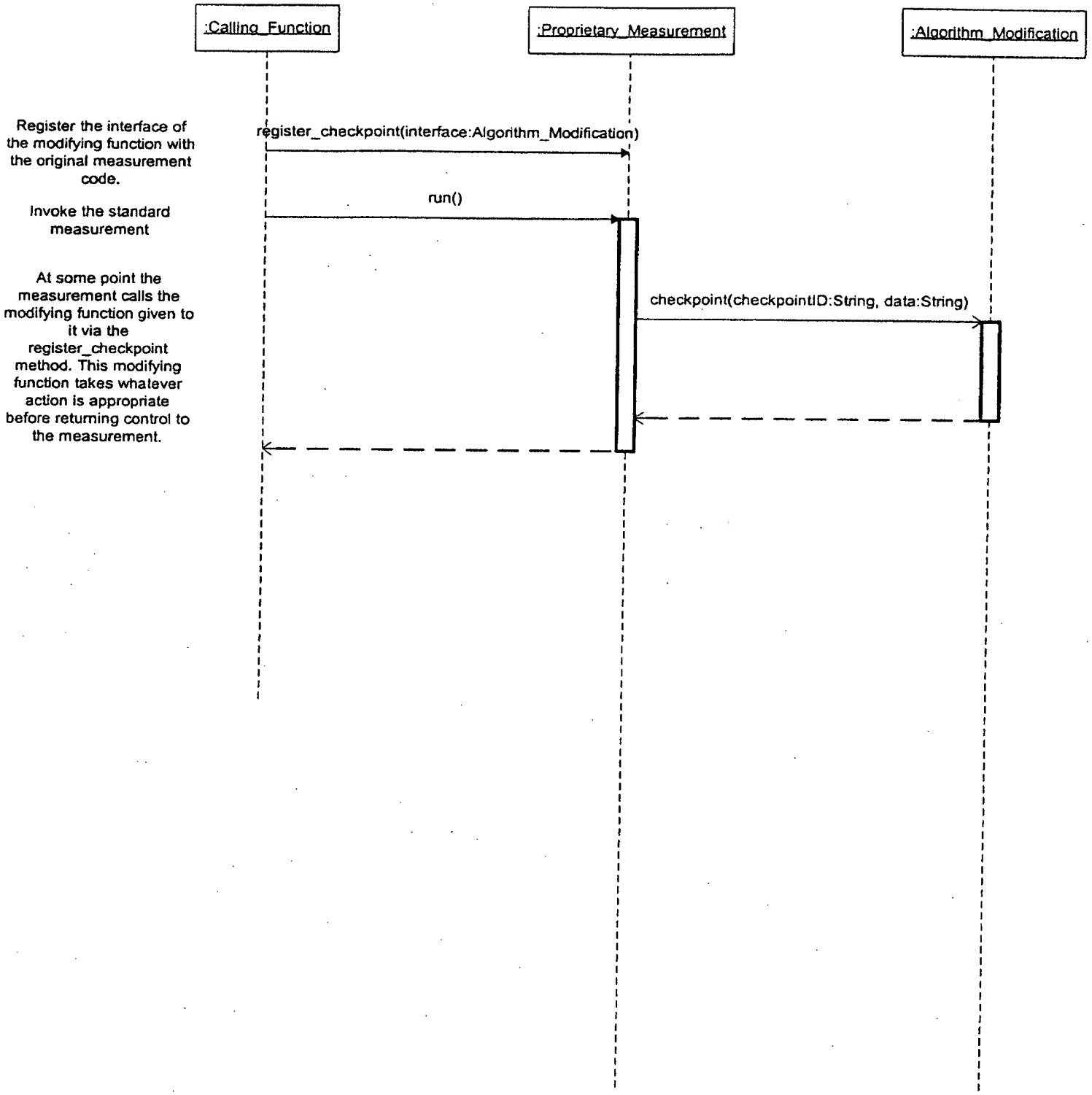
Description of the Invention

A measurement will include one or more "variation points" in its algorithm. These are points in the code where the designer anticipates that the user might wish to modify or interact with the algorithm. A variation point will contain a "call-out" (often called an outgoing interface) to user supplied code. The call-out will contain information about the current state of the measurement. The user can take any required action, such as inserting a short delay, controlling the Device Under Test, or modifying the existing measurement algorithm. The user code then returns to the measurement algorithm for normal processing.

There are several possible variations on the technology for performing the call-out and the type of parameters and control information communicated. Some of the call-out variations are: calls into well-known functions in defined DLL's, Connection-Point interfaces (Visual Basic "Events"), an event server, instantiation of defined COM server interfaces, invocation of COM interfaces passed in to the measurement algorithm, use of CORBA-based servers, use of Enterprise Java Beans as an interface mechanism, use of SOAP or Microsoft web services as the interface mechanism.

Some of the variants of the control information are: each variation point calls a separate interface, each variation point calls the same interface and passes a parameter to describe the variation point location, a variation point call-out can contain a "cancel" parameter to allow the user to override a decision or cancel an operation, a variation point can contain instruction codes to be sent to an instrument to allow user edit or optimization, and a variation point can pass codes to allow the user to select options or algorithms from several alternatives offered by the measurement algorithm.


«interface»
ICheckpoint
+checkpoint(in checkpointID : String, in data : String) : String
Measurement
+register_checkpoint(in interface : *Algorithm_Modification)
+run()
ICheckpoint
Algorithm_Modification
+checkpoint(in checkpointID : String, in data : String) : String


:Calling_Function
:Proprietary_Measurement
:Algorithm_Modification
Register the interface of the modifying function with the original measurement code.
register_checkpoint(interface:Algorithm_Modification)
Invoke the standard measurement
run()
At some point the measurement calls the modifying function given to it via the register_checkpoint method. This modifying function takes whatever action is appropriate before returning control to the measurement.
checkpoint(checkpointID:String, data:String)